Strong Rules for Discarding Predictors in Lasso-type Problems

Robert Tibshirani †
Jacob Bien
Jerome Friedman
Trevor Hastie
Noah Simon
Jonathan Taylor
Ryan J. Tibshirani

Departments of Statistics and Health Research and Policy, Stanford University, Stanford CA 94305, USA. Email: tibs@stanford.edu

July 28, 2011

Summary. We consider rules for discarding predictors in lasso regression and related problems, for computational efficiency. El Ghaoui et al. (2010) propose “SAFE” rules, based on univariate inner products between each predictor and the outcome, that guarantee a coefficient will be zero in the solution vector. This provides a reduction in the number of variables that need to be entered into the optimization. In this paper, we propose *strong rules* that are very simple and yet screen out far more predictors than the SAFE rules. This great practical improvement comes at a price: the strong rules are not foolproof and can mistakenly discard active predictors, that is, ones that have nonzero coefficients in the solution. We therefore combine them with simple checks of the Karush-Kuhn-Tucker (KKT) conditions to ensure that the exact solution to the convex problem is delivered. Of course, any (approximate) screening method can be combined with the KKT conditions to ensure the exact solution; the strength of the strong rules lies in the fact that, in practice, they discard a very large number of the inactive predictors and almost never commit mistakes. We also derive conditions under which they are foolproof. Strong rules provide a substantial savings in computational time for a variety of statistical optimization problems.

1. Introduction

Our focus here is statistical models fit using ℓ_1 penalization, starting with penalized linear regression. Consider a problem with N observations and p predictors. Let $\mathbf{y}$ denote the N-vector of outcomes, and $\mathbf{X}$ be the $N \times p$ matrix of predictors, with ith row x_i and jth column $\mathbf{x}_j$. For a set of indices $\mathcal{A} = \{j_1, \ldots j_k\}$, we write $\mathbf{X}_{\mathcal{A}}$ to denote the $N \times k$ submatrix $\mathbf{X}_{\mathcal{A}} = [\mathbf{x}_{j_1}, \ldots \mathbf{x}_{j_k}]$, and we write $\mathbf{b}_{\mathcal{A}} = (b_{j_1}, \ldots b_{j_k})$ for a vector $\mathbf{b}$. We assume that the predictors and outcome have been centered, so that we can omit an intercept term from the model. The lasso (Tibshirani (1996), Chen et al. (1998)) optimization problem is

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{1}$$

†Corresponding author.

where $\lambda \geq 0$ is a tuning parameter.

There has been considerable work in the past few years deriving fast algorithms for this problem, especially for large values of N and p. A main reason for using the lasso is that the ℓ_1 penalty tends to set some entries of $\hat{\boldsymbol{\beta}}$ to exactly zero, and therefore it performs a kind of variable selection. Now suppose we knew, a priori to solving (1), that a subset of the variables $\mathcal{D} \subseteq \{1, \dots p\}$ will be inactive at the solution, that is, they will have zero coefficients: $\hat{\boldsymbol{\beta}}_{\mathcal{D}} = 0$. ‡ Then we could discard the variables in $\mathcal{D}$ from the optimization, replacing the design matrix in (1) by $\mathbf{X}_{\mathcal{D}^c}$, $\mathcal{D}^c = \{1, \dots p\} \setminus \mathcal{D}$, and just solve for the remaining coefficients $\hat{\boldsymbol{\beta}}_{\mathcal{D}^c}$. For a relatively large set $\mathcal{D}$, this would result in a substantial computational savings.

El Ghaoui et al. (2010) construct such a set of discarded variables by looking at the univariate inner products of each predictor with the response. Namely, their "SAFE" rule discards the jth variable if

$$|\mathbf{x}_j^T \mathbf{y}| < \lambda - \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \frac{\lambda_{\max} - \lambda}{\lambda_{\max}}, \tag{2}$$

where $\lambda_{\max} = \max_i |\mathbf{x}_i^T \mathbf{y}|$ is the smallest tuning parameter value for which all coefficients in the solution are zero. In deriving this rule, the authors prove that any predictor satisfying (2) must be inactive at the solution; said differently, condition (2) implies that $\hat{\beta}_j = 0$. (Their proof relies on the dual of problem (1); it has nothing to do with the rest of this paper, but we summarize it in the Appendix because we find it interesting.) The authors then show that applying the SAFE rule (2) to discard predictors can save both time and memory in the overall computation, and also derive analogous rules for ℓ_1-penalized logistic regression and ℓ_1-penalized support vector machines.

The existence of any such rule is surprising (at least to us), and the work presented here was inspired by the SAFE work. In this paper, we propose *strong rules* for discarding predictors in the lasso and other problems that involve lasso-type penalties. The basic strong rule for the lasso looks like a modification of (2), with $\|\mathbf{x}_j\|_2 \|\mathbf{y}\|_2 / \lambda_{\max}$ replaced by 1: it discards the jth variable if

$$|\mathbf{x}_j^T \mathbf{y}| < \lambda - (\lambda_{\max} - \lambda) = 2\lambda - \lambda_{\max}. \tag{3}$$

The strong rule (3) tends to discard more predictors than the SAFE rule (2). For standardized predictors ($\|\mathbf{x}_j\|_2 = 1$ for all j), this will always be the case, as $\|\mathbf{y}\|_2 / \lambda_{\max} \geq 1$ by the Cauchy–Schwartz inequality. However, the strong rule (3) can erroneously discard active predictors, ones that have nonzero coefficients in the solution. Therefore we rely on the Karush-Kuhn-Tucker (KKT) conditions to ensure that we are indeed computing the correct coefficients in the end. A simple strategy would be to add the variables that fail a KKT check back into the optimization. We discuss more sophisticated implementation techniques, specifically in the context of our `glmnet` algorithm, in Section 7 at the end of the paper.

The most important contribution of this paper is a version of the strong rules that can be used when solving the lasso and lasso-type problems over a grid of tuning parameter values $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$. We call these the *sequential strong rules*. For the lasso, having

‡If $\mathbf{X}$ does not have full column rank, which is necessarily the case when $p > N$, then there may not be a unique lasso solution; we don't pay special attention to this case, and will write "the solution" when we really mean "a solution".

already computed the solution $\hat{\boldsymbol{\beta}}(\lambda_{k-1})$ at λ_{k-1}, the sequential strong rule discards the jth predictor from the optimization problem at λ_k if

$$\left|\mathbf{x}_j^T\big(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}(\lambda_{k-1})\big)\right| < 2\lambda_k - \lambda_{k-1}. \tag{4}$$

The sequential rule (4) performs much better than both the basic rule (3) and the SAFE rule (2), as we demonstrate in Section 2. El Ghaoui et al. (2011) also propose a version of the SAFE rule that can be used when considering multiple tuning parameter values, called "recursive SAFE", but it too is clearly outperformed by the sequential strong rule. Like its basic counterpart, the sequential strong rule can mistakenly discard active predictors, so it must be combined with a check of the KKT conditions (see Section 7 for details).

At this point, the reader may wonder: any approximate or non-exact rule for discarding predictors can be combined with a check of the KKT conditions to ensure the exact solution—so what makes the sequential strong rule worthwhile? Our answer is twofold:

(a) In practice, the sequential strong rule is able to discard a very large proportion of inactive predictors, and rarely commits mistakes by discarding active predictors. In other words, it serves as a very effective heuristic.
(b) The motivating arguments behind the sequential strong rule are quite simple and the same logic can be used to derive rules for ℓ_1-penalized logistic regression, the graphical lasso, the group lasso, and others.

The mistakes mentioned in (a) are so rare that for a while a group of us were trying to prove that the sequential strong rule for the lasso was foolproof, while others were trying to find counterexamples (hence the large number of coauthors!). We finally did find some counterexamples of the sequential strong rule and one such counterexample is given in Section 3, along with some analysis of rule violations in the lasso case. Furthermore, despite the similarities in appearance of the basic strong rule (3) to the SAFE rule (2), the arguments motivating the strong rules (3) and (4) are entirely different, and rely on a simple underlying principle. In Section 4 we derive analogous rules for the elastic net, and in Section 5 we derive rules for ℓ_1-penalized logistic regression. We give a version for more general convex problems in Section 6, covering the graphical lasso and group lasso as examples.

Finally, we mention some related work. Wu et al. (2009) study ℓ_1 penalized logistic regression and build a screened set $\mathcal{S}$ based on the inner products between the outcome and each feature. As with the strong rules, their construction does not guarantee that the variables in $\mathcal{S}$ actually have zero coefficients in the solution, and so after fitting on $\mathbf{X}_{\mathcal{S}^c}$, the authors check the KKT optimality conditions for violations. In the case of violations, they weaken their set $\mathcal{S}$, and repeat this process. Also, Fan & Lv (2008) study the screening of variables based on their inner products in the lasso and related problems, but not from an optimization point of view; their screening rules may again set coefficients to zero that are nonzero in the solution, however, the authors argue that under certain situations this can lead to better performance in terms of estimation risk.

2. Strong rules for the lasso

2.1. Definitions and simulation studies

As defined in the introduction, the basic strong rule for the lasso discards the jth predictor from the optimization problem if

$$|\mathbf{x}_j^T\mathbf{y}| < 2\lambda - \lambda_{\max}, \tag{5}$$

where $\lambda_{\max} = \max_j |\mathbf{x}_j^T \mathbf{y}|$ is the smallest tuning parameter value such that $\hat{\boldsymbol{\beta}}(\lambda_{\max}) = 0$. If we are interested in the solution at many values $\lambda_1 \geq \ldots \geq \lambda_m$, then having computed the solution $\hat{\boldsymbol{\beta}}(\lambda_{k-1})$ at λ_{k-1}, the sequential strong rule discards the jth predictor from the optimization problem at λ_k if

$$\left|\mathbf{x}_j^T \left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}(\lambda_{k-1})\right)\right| < 2\lambda_k - \lambda_{k-1}. \tag{6}$$

Here we take $\lambda_0 = \lambda_{\max}$. As $\hat{\boldsymbol{\beta}}(\lambda_{\max}) = 0$, the basic strong rule (5) is a special case of the sequential rule (6).

First of all, how does the basic strong rule compare to the basic SAFE rule (2)? When the predictors are standardized (meaning that $\|\mathbf{x}_i\|_2 = 1$ for every i), it is easy to see that the basic strong bound is always larger than the basic SAFE bound, because $\|\mathbf{y}\|_2/\lambda_{\max} \geq 1$ by the Cauchy-Schwartz inequality. When the predictors are not standardized, the ordering between the two bounds is not as clear, but in practice the basic strong rule still tends to discard more predictors unless the marginal variances of the predictors are wildly different (by factors of say 10 or more). Figure 1 demonstrates the bounds for a simple example.

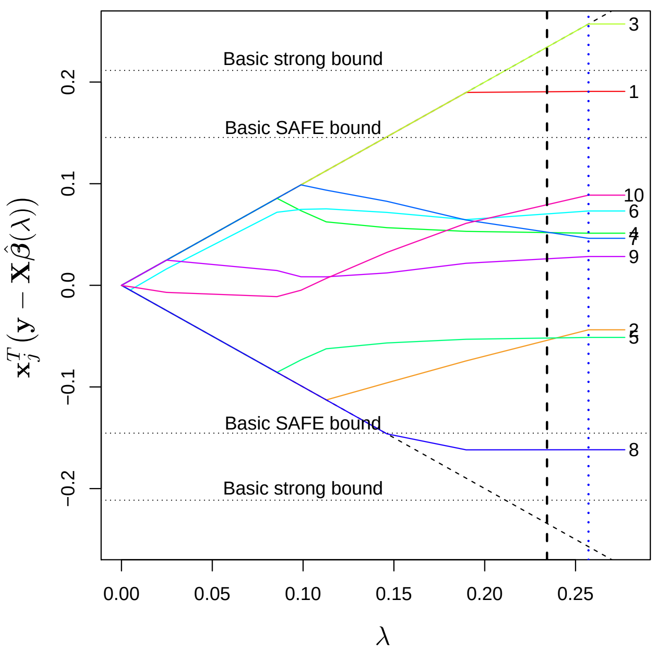

Fig. 1. *Basic SAFE and basic strong bounds in a simple example with 10 predictors, labelled at the right. The plot shows the inner product of each predictor with the current residual, $\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\hat{\beta}(\lambda))$, as a function of λ. The predictors that are in the model are those with maximal (absolute) inner product, equal to $\pm\lambda$. The dotted vertical line is drawn at $\lambda_{\max}$; the dashed vertical line is drawn at some value $\lambda = \lambda'$ at which we want to discard predictors. The basic strong rule keeps only predictor number 3, while the basic SAFE rule keeps predictors 8 and 1 as well.*

More importantly, how do the rules perform in practice? Figures 2 and 3 attempt to answer this question by examining several simulated data sets. (A few real data sets are considered later in Section 3.2.) In Figure 2, we compare the performance of the basic

SAFE rule, recursive SAFE rule, basic strong rule, and sequential strong rule in discarding predictors for the lasso problem along a sequence of 100 tuning parameter values, equally spaced on the log scale. The three panels correspond to different scenarios for the model matrix $\mathbf{X}$; in each we plot the number of active predictors in the lasso solution on the x-axis, and the number of predictors left after filtering with the proposed rules (i.e. after discarding variables) on the y-axis. Shown are the basic SAFE rule, the recursive SAFE rule, the global strong rule and the sequential strong rule. The details of the data generation are given in the Figure caption. The sequential strong rule is remarkably effective.

It is common practice to standardize the predictors before applying the lasso, so that the penalty term makes sense. This is what was done in the examples of Figure 2. But in some instances, one might not want to standardize the predictors, and so in Figure 3 we investigate the performance of the rules in this case. In the left panel the population variance of each predictor is the same; in the right panel it varies by a factor of 50. We see that in the latter case the SAFE rules outperform the basic strong rule, but the sequential strong rule is still the clear winner. There were no violations of either of the strong rules in either panel.

After seeing the performance of the sequential strong rule, it might seem like a good idea to combine the basic SAFE rule with the sequential strategy; this yields the *sequential SAFE rule*, which discards the jth predictor at the parameter value λ_k if

$$\left|\mathbf{x}_j^T\left(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}(\lambda_{k-1})\right)\right| < \lambda_k - \|\mathbf{x}_j\|_2\|\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}(\lambda_{k-1})\|_2\frac{\lambda_{k-1}-\lambda_k}{\lambda_{k-1}}. \tag{7}$$

We believe that this rule is not foolproof, in the same way that the sequential strong rule is not foolproof, but have not yet found an example in which it fails. In addition, while (7) outperforms the basic and recursive SAFE rules, we have found that it is not nearly as effective as the sequential strong rule at discarding predictors and hence we do not consider it further.

2.2. Motivation for the strong rules

We now give some motivation for the sequential strong rule (6). The same motivation also applies to the basic strong rule (5), recalling that the basic rule corresponds to the special case $\lambda_0 = \lambda_{\max}$ and $\hat{\boldsymbol{\beta}}(\lambda_{\max}) = 0$.

We start with the KKT conditions for the lasso problem (1). These are

$$\mathbf{x}_j^T(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}) = \lambda\gamma_j \quad \text{for} \ \ j=1,\ldots p, \tag{8}$$

where γ_j is the jth component of the subgradient of $\|\hat{\boldsymbol{\beta}}\|_1$:

$$\gamma_j \in \begin{cases} \{+1\} & \text{if } \ \hat{\beta}_j > 0 \\ \{-1\} & \text{if } \ \hat{\beta}_j < 0 \\ [-1,1] & \text{if } \ \hat{\beta}_j = 0. \end{cases} \tag{9}$$

Let $c_j(\lambda) = \mathbf{x}_j^T(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}(\lambda))$, where we emphasize the dependence on λ. The key idea behind the strong rules is to assume that each $c_j(\lambda)$ is non-expansive in λ, that is,

$$|c_j(\lambda)-c_j(\tilde{\lambda})| \le |\lambda-\tilde{\lambda}| \quad \text{for any} \ \ \lambda,\tilde{\lambda}, \ \ \text{and} \ \ j=1,\ldots p. \tag{10}$$

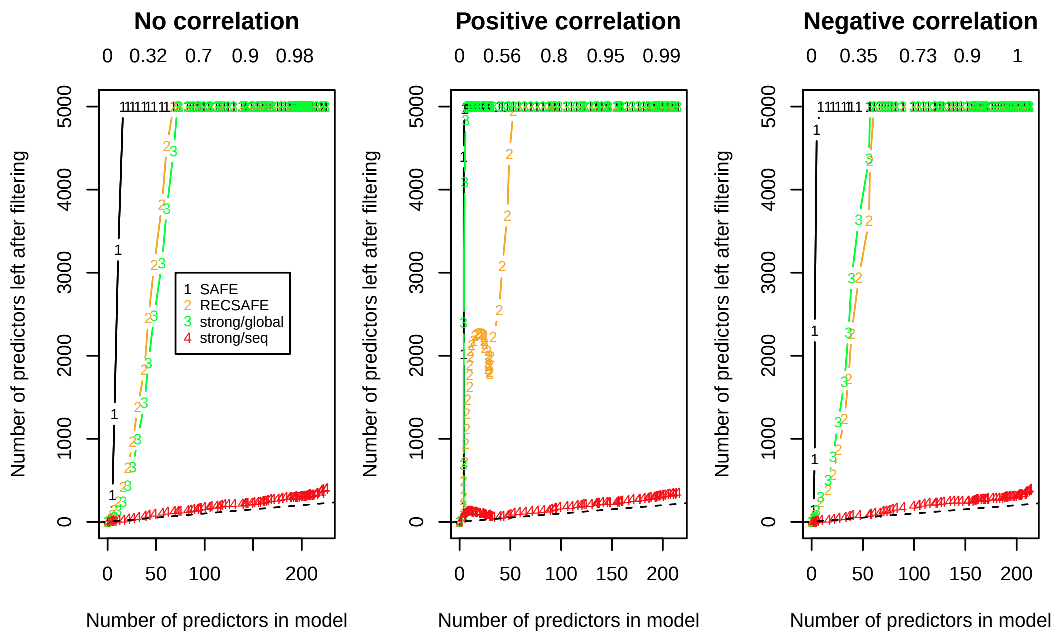

Fig. 2. Lasso regression: results of different rules applied to three different scenarios. Shown are the number of predictors left after screening at each stage, plotted against the number of predictors in the model for a given value of λ. The value of λ is decreasing as we move from left to right. There are three scenarios with various values of N and p; in the first two panels the $\mathbf{X}$ matrix entries are i.i.d. standard Gaussian with pairwise correlation zero (left), and 0.5 (middle). In the right panel, one quarter of the pairs of features (chosen at random) had correlation -0.8. In the plots, we are fitting along a path of 100 decreasing λ values equally spaced on the log-scale, A broken line with unit slope is added for reference. The proportion of variance explained by the model is shown along the top of the plot. There were no violations of either of the strong rules any of the three scenarios.

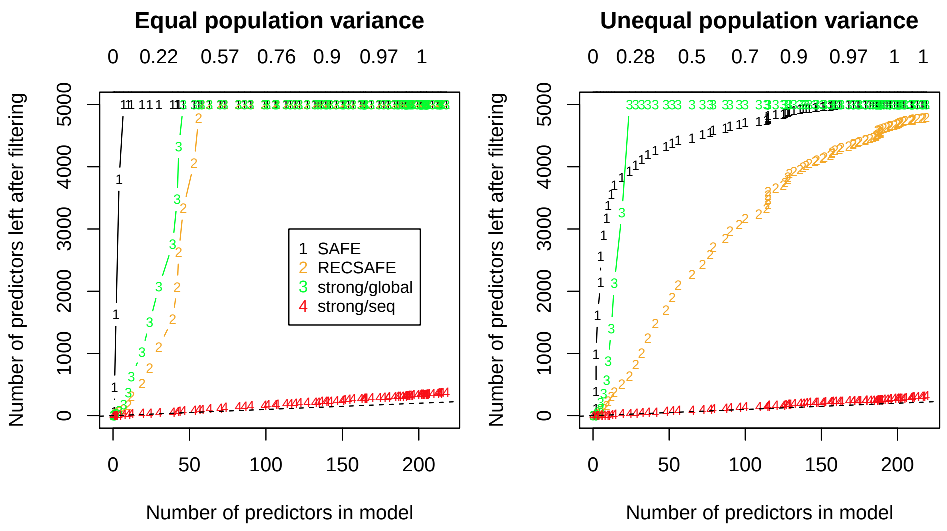

Fig. 3. *Lasso regression: results of different rules when the predictors are not standardized. The scenario in the left panel is the same as in the top left panel of Figure 2, except that the features are not standardized before fitting the lasso. In the data generation for the right panel, each feature is scaled by a random factor between 1 and 50, and again, no standardization is done.*

This condition is equivalent to $c_j(\lambda)$ being differentiable almost everywhere, and satisfying $|c_j'(\lambda)| \leq 1$ wherever this derivative exists, for $j = 1, \dots p$. Hence we call (10) the "unit slope" bound.

Using condition (10), if we have $|c_j(\lambda_{k-1})| < 2\lambda_k - \lambda_{k-1}$, then

$$\begin{aligned} |c_j(\lambda_k)| &\leq |c_j(\lambda_k) - c_j(\lambda_{k-1})| + |c_j(\lambda_{k-1})| \\ &< (\lambda_{k-1} - \lambda_k) + (2\lambda_k - \lambda_{k-1}) \\ &= \lambda_k, \end{aligned}$$

which implies that $\hat{\beta}_j(\lambda_k) = 0$ by the KKT conditions (8) and (9). But this is exactly the sequential strong rule (6), because $c_j(\lambda_k) = \mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}(\lambda_k))$. In words: assuming that we can bound the amount that $c_j(\lambda)$ changes as we move from λ_{k-1} to λ_k, if the initial inner product $c_j(\lambda_{k-1})$ is too small, then it cannot "catch up" in time. An illustration is given in Figure 4.

The arguments up until this point do not really depend on the Gaussian lasso problem in any critical way, and similar arguments can be made to derive strong rules for ℓ_1-penalized logistic regression and more general convex problems. But in the specific context of the lasso, the strong rules, and especially the unit slope assumption (10), can be explained more concretely. For simplicity, the arguments provided here assume that $\text{rank}(\mathbf{X}) = p$, so that necessarily $p \leq N$, although similar arguments can be used motivate the $p > N$ case. Let $\mathcal{A}$ denote the set of active variables in the lasso solution,

$$\mathcal{A} = \{j : \hat{\beta}_j \neq 0\}.$$

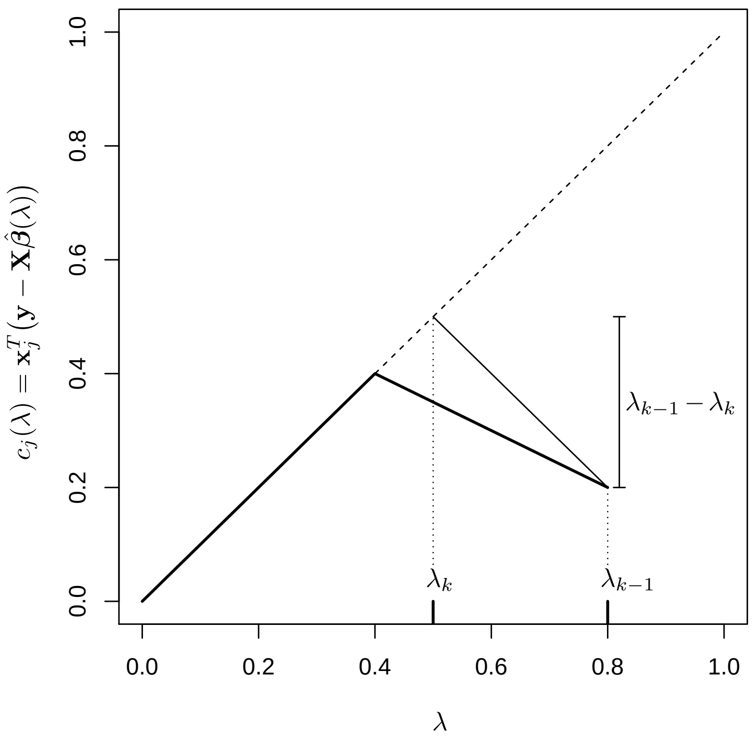

Fig. 4. *Illustration of the slope bound* (10) *leading to the strong rules* (5) *and* (6). *The inner product (thick line) c_j is plotted in as a function of λ, restricted to only one predictor for simplicity. The slope of c_j between λ_{k-1} and λ_k is bounded in absolute value by 1, so the most it can rise over this interval is $\lambda_{k-1} - \lambda_k$. Therefore, if it starts below $\lambda_k - (\lambda_{k-1} - \lambda_k) = 2\lambda_k - \lambda_{k-1}$, it cannot possibly reach the critical level by λ_k.*

Also let $s = \text{sign}(\hat{\boldsymbol{\beta}}_\mathcal{A})$. Note that $\mathcal{A}, s$ are implicitly functions of λ. It turns out that we can express the lasso solution entirely in terms of $\mathcal{A}$ and s:

$$\hat{\boldsymbol{\beta}}_\mathcal{A}(\lambda) = (\mathbf{X}_\mathcal{A}^T\mathbf{X}_\mathcal{A})^{-1}(\mathbf{X}_\mathcal{A}^T y - \lambda s) \tag{11}$$

$$\hat{\boldsymbol{\beta}}_{\mathcal{A}^c}(\lambda) = 0, \tag{12}$$

where we write $\mathbf{X}_\mathcal{A}^T$ to mean $(\mathbf{X}_\mathcal{A})^T$. On an interval of λ in which the active set doesn't change, the solution (11), (12) is just linear in λ. Also, the solution (11), (12) is continuous at all values of λ at which the active set does change. (For a reference, see Efron et al. (2004).) Therefore the lasso solution is a continuous, piecewise linear function of λ, as is $c_j(\lambda) = \mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}(\lambda))$. The critical points, or changes in slope, occur whenever a variable enters or leaves the active set. Each $c_j(\lambda)$ is differentiable at all values of λ that are not critical points, which means it is differentiable almost everywhere (since the set of critical points is countable and hence has measure zero). Further, $c_j'(\lambda)$ is just the slope of the piecewise linear path at λ, and hence (10) is really just a slope bound. By expanding (11),

(12) in the definition of $c_j(\lambda)$, it is not hard to see that the slope at λ is

$$c_j'(\lambda) = \begin{cases} s_j & \text{for } \ j \in \mathcal{A} \\ \mathbf{x}_j^T \mathbf{X}_{\mathcal{A}} (\mathbf{X}_{\mathcal{A}}^T \mathbf{X}_{\mathcal{A}})^{-1} s & \text{for } \ j \notin \mathcal{A}. \end{cases} \tag{13}$$

Therefore the slope condition $|c_j'(\lambda)| \leq 1$ is satisfied for all active variables $j \in \mathcal{A}$. For inactive variables it can fail, but is unlikely to fail if the correlation between the variables in $\mathcal{A}$ and $\mathcal{A}^c$ is small (thinking of standardized variables). From (13), we can rewrite the slope bound (10) as

$$\|\mathbf{X}_{\mathcal{A}^c}^T \mathbf{X}_{\mathcal{A}} (\mathbf{X}_{\mathcal{A}}^T \mathbf{X}_{\mathcal{A}})^{-1} \text{sign}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}(\lambda))\|_\infty \leq 1 \quad \text{for all } \ \lambda. \tag{14}$$

In this form, the condition looks like the well-known "irrepresentable condition", which we discuss in the next section.

2.3. Connection to the irrepresentable condition

A common condition appearing in work about model selection properties of lasso is the "irrepresentable condition" Zhao & Yu (2006), Wainwright (2009), Candes & Plan (2009), which is closely related to the concept of "mutual incoherence" Fuchs (2005), Tropp (2006), Meinhausen & Buhlmann (2006). If $\mathcal{T}$ is the set of variables present in the true (underlying) linear model, that is

$$\mathbf{y} = \mathbf{X}_{\mathcal{T}} \boldsymbol{\beta}_{\mathcal{T}} + \mathbf{z}$$

where $\boldsymbol{\beta}_{\mathcal{T}} \in \mathbb{R}^{|\mathcal{T}|}$ is the true coefficient vector and $\mathbf{z} \in \mathbb{R}^n$ is noise, then the irrepresentable condition is that

$$\|\mathbf{X}_{\mathcal{T}^c}^T \mathbf{X}_{\mathcal{T}} (\mathbf{X}_{\mathcal{T}}^T \mathbf{X}_{\mathcal{T}})^{-1} \text{sign}(\boldsymbol{\beta}_{\mathcal{T}})\|_\infty \leq 1 - \epsilon \tag{15}$$

for some $0 < \epsilon \leq 1$.

The conditions (15) and (14) appear extremely similar, but a key difference between the two is that the former pertains to the true coefficients generating the data, while the latter pertains to those found by the lasso optimization problem. Because $\mathcal{T}$ is associated with the true model, we can put a probability distribution on it and a probability distribution on sign$(\boldsymbol{\beta}_{\mathcal{T}})$, and then show that with high probability, certain design matrices $\mathbf{X}$ satisfy (15). For example, Candes & Plan (2009) show that if $|\mathcal{T}|$ is small, $\mathcal{T}$ is drawn from the uniform distribution on $|\mathcal{T}|$-sized subsets of $\{1, \ldots p\}$, and each entry of sign$(\boldsymbol{\beta}_{\mathcal{T}})$ is equal to ± 1 with equal probability, then designs $\mathbf{X}$ with $\max_{j \neq k} |\mathbf{x}_j^T \mathbf{x}_k| = O(1/\log p)$ satisfy the irrepresentable condition (15) with very high probability. Unfortunately the same types of arguments cannot be applied directly to (14). A distribution on $\mathcal{T}$ and sign$(\boldsymbol{\beta}_{\mathcal{T}})$ induces a different distribution on $\mathcal{A}$ and sign$(\hat{\boldsymbol{\beta}}_{\mathcal{A}})$, via the lasso optimization procedure. Even if the distributions of $\mathcal{T}$ and sign$(\boldsymbol{\beta}_{\mathcal{T}})$ are very simple, the distributions of $\mathcal{A}$ and sign$(\hat{\boldsymbol{\beta}}_{\mathcal{A}})$ are likely to be complicated.

Under the same assumptions as those described above, and an additional assumption that the signal-to-noise ratio is high, Candes & Plan (2009) prove that for $\lambda = 2\sqrt{2 \log p}$ the lasso solution satisfies

$$\mathcal{A} = \mathcal{T} \quad \text{and} \quad \text{sign}(\hat{\boldsymbol{\beta}}_{\mathcal{A}}) = \text{sign}(\boldsymbol{\beta}_{\mathcal{T}})$$

with high probability. In this event, conditions (14) and (15) identical; therefore the work of Candes & Plan (2009) proves that (14) also holds with high probability, under the stated assumptions and only when $\lambda = 2\sqrt{2\log p}$. For our purposes, this is not incredibly useful because we want the slope bound to hold along the entire path, that is, for all λ. But still, it seems reasonable that confidence in (15) should translate to some amount of confidence in (14). And luckily for us, we do not need the slope bound (14) to hold exactly or with any specified level of probability, because we are using it as a computational tool and revert to checking the KKT conditions when it fails.

3. Violations of the strong rules

3.1. A simple counterexample

Here we demonstrate a counterexample of both the slope bound (10) and the sequential strong rule (6). We chose $N = 50$ and $p = 30$, with the entries of $\mathbf{y}$ and $\mathbf{X}$ drawn independently from a standard normal distribution. Then we centered $\mathbf{y}$ and the columns of $\mathbf{X}$, and scaled the columns of $\mathbf{X}$ to have unit norm. As Figure 5 shows, for predictor $j = 2$, the slope of $c_j(\lambda) = \mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}(\lambda))$ is $c_j'(\lambda) = -1.586$ for all $\lambda \in [\lambda_2, \lambda_1]$, where $\lambda_2 = 0.0244$, $\lambda_1 = 0.0259$. Moreover, if we were to use the solution at λ_1 to eliminate predictors for the fit at λ_2, then we would eliminate the 2nd predictor based on the bound (6). But this is clearly a problem, because the 2nd predictor enters the model at λ_2. By continuity, we can choose λ_2 in an interval around 0.0244 and λ_1 in an interval around 0.0259, and still break the sequential strong rule (6).

We believe that a counterexample of the basic strong rule (5) can also be constructed, but we have not yet found one. Such an example is somewhat more difficult to construct because it would require that the average slope exceed 1 from $\lambda_{\max}$ to λ, rather than exceeding 1 for short stretches of λ values.

3.2. Numerical investigation of violations

We generated Gaussian data with $N = 100$, and the number predictors p varying over the set $\{20, 50, 100, 500, 1000\}$. The predictors had pairwise correlation 0.5. (With zero pairwise correlation, $\mathbf{X}^T\mathbf{X}$ would be orthogonal in the population and hence "close to" orthogonal in the sample, making it easier for the strong rules to hold—see the next section. Therefore we chose pairwise correlation 0.5 in order to challenge the rules.) For each value of p, we chose one quarter variables uniformly at random, assigned them coefficient values equal to ± 2 with equal probability, and added Gaussian noise to the true signal to generate $\mathbf{y}$. Then we standardized $\mathbf{y}$ and the columns of $\mathbf{X}$. We ran the R package `glmnet` v1.5, which uses a path of 100 values of λ spanning the entire operating range, equally spaced on a log scale. This was used to determine the exact solutions, and then we recorded the number of violations of the sequential strong rule.

Figure 6 shows the results averaged over 100 draws of the simulated data. We plot the percent variance explained on the x-axis (instead of λ, since the former is more meaningful), and the total number of violations (out of p predictors) the y-axis. We see that violations are quite rare, in general never averaging more than 0.3 erroneously discarded predictors! They are more common at the unregularized (small λ) end of the path and also tend to occur when p is fairly close to N.§ When $p \gg N$ ($p = 500$ or 1000 here), there were no violations in any of 100 the simulated data sets. It is perhaps not surprisingly, then, that

§When $p = N$, the model is able to produce a saturated fit, but only "just". So for this scenario, the coefficient paths are somewhat erratic near the end of the path.

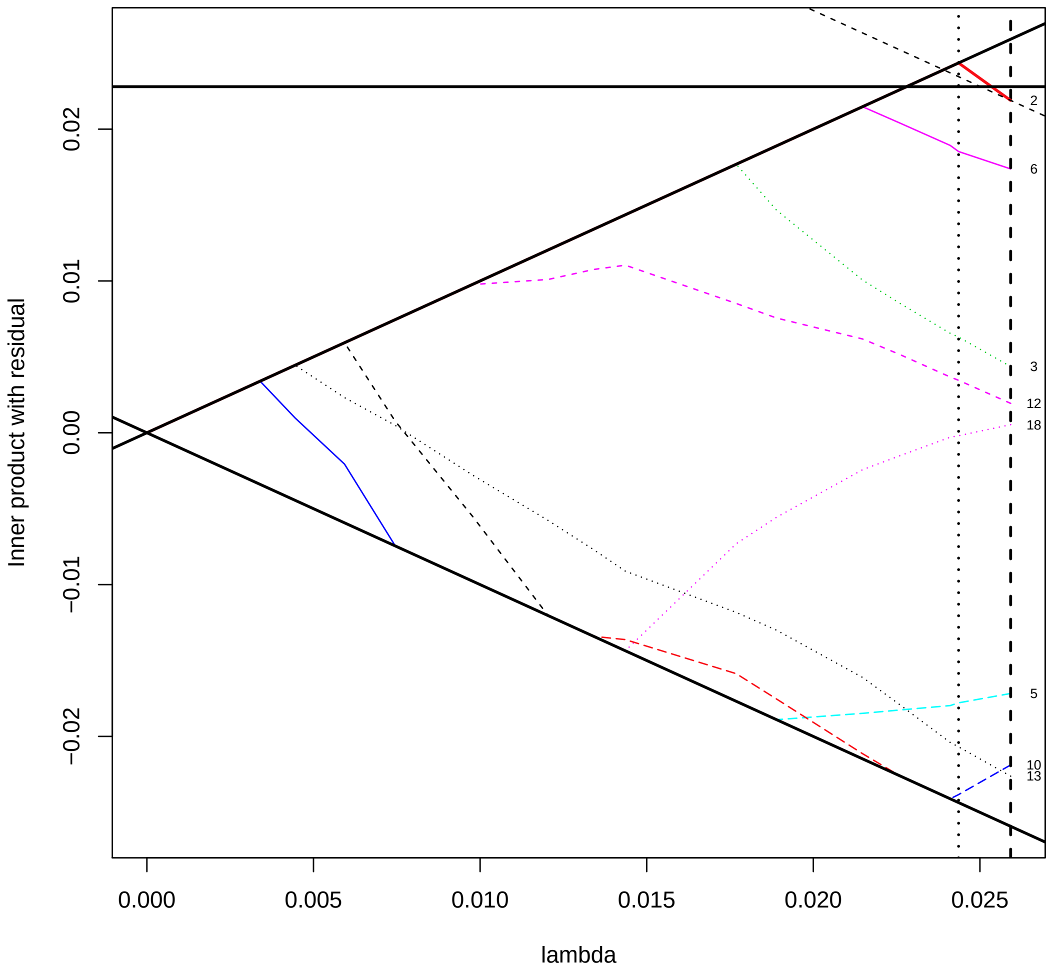

Fig. 5. *Example of a violation of the slope bound* (10)*, which breaks the sequential strong rule* (6)*. The entries of* $\mathbf{y}$ *and* $\mathbf{X}$ *were generated as independent, standard normal random variables with* $N = 50$ *and* $p = 30$*. (Hence there is no underlying signal.) The lines with slopes* $\pm\lambda$ *are the envelopes of maximal inner products achieved by predictors in the model for each* λ*. For clarity we only show a short stretch of the solution path. The dashed vertical line is drawn at* λ_1*, and we are considering the the solution at a new value* $\lambda_2 < \lambda_1$*, the dotted vertical line to its left. The dotted horizontal line is the bound* (6)*. In the top right part of the plot, the inner product path for the predictor* $j = 2$ *and starts below the bound, but enters the model at* λ_2*. The slope of path for predictor 2 between* λ_1 *and* λ_2 *is -1.586. A broken line of slope -1 is drawn beside it for reference. The plot contains other examples of large slopes leading to rule violations, for example, around* $\lambda = 0.007$*.*

there were no violations in the examples shown in Figures 2 and 3 since there we had $p \gg N$ as well.

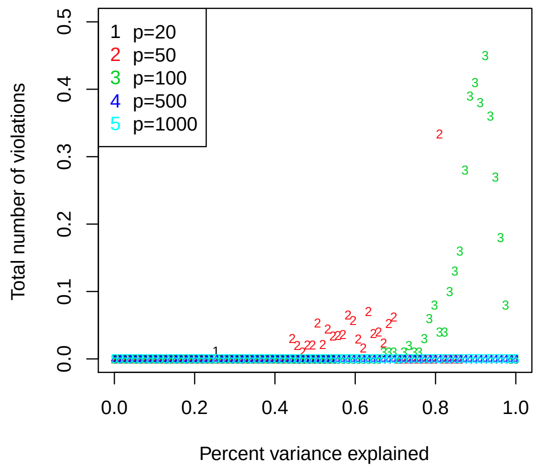

Fig. 6. *Total number of violations (out of p predictors) of the sequential strong rule, for simulated data with $N = 100$ and different values of p. A sequence of models is fit, over 100 decreasing values of λ as we move from left to right. The features are drawn from a Gaussian distribution with pairwise correlation 0.5. The results are averages over 100 draws of the simulated data.*

In Table 1 we applied the strong rules to three large datasets from the UCI machine learning repository, and a standard microarray dataset. As before, we applied `glmnet` along a path of about 100 values of λ values. There were no violations of the rule in any of the solution paths, and a large fraction of the predictors were successfully discarded. We investigate the computational savings that result from the strong rule in Section 7.

3.3. A sufficient condition for the slope bound

Tibshirani & Taylor (2011) prove a general result that can be used to give the following sufficient condition for the unit slope bound (10). Under this condition, both basic and sequential strong rules will never discard active predictors. Recall that an $m \times m$ matrix $\mathbf{A}$ is diagonally dominant if $|A_{ii}| \geq \sum_{j \neq i} |A_{ij}|$ for all $i = 1, \ldots m$. Their result gives us the following:

THEOREM 1. *Suppose that* $\mathbf{X}$ *has full column rank, that is,* $\text{rank}(X) = p$. *If*

$$(\mathbf{X}^T\mathbf{X})^{-1} \text{ is diagonally dominant,} \tag{16}$$

then the slope bound (10) *holds, and hence the strong rules* (5), (6) *never produce violations.*

PROOF. Tibshirani & Taylor (2011) consider a problem

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\text{argmin}} \; \frac{1}{2}\|\mathbf{y} - \boldsymbol{\alpha}\|_2^2 + \lambda\|\mathbf{D}\boldsymbol{\alpha}\|_1, \tag{17}$$

where $\mathbf{D}$ is a general $m \times n$ penalty matrix. They derive the dual problem corresponding to (17), which has a dual solution $\hat{\mathbf{u}}(\lambda)$ relating to the primal solution $\hat{\boldsymbol{\alpha}}(\lambda)$ by

$$\hat{\boldsymbol{\alpha}}(\lambda) = \mathbf{y} - \mathbf{D}^T \hat{\mathbf{u}}(\lambda).$$

In the proof of their "boundary lemma", Lemma 1, they show that if $\mathbf{D}\mathbf{D}^T$ is diagonally dominant, then the dual solution satisfies

$$|\hat{u}_j(\lambda) - \hat{u}_j(\tilde{\lambda})| \leq |\lambda - \tilde{\lambda}| \quad \text{for any} \;\; \lambda, \tilde{\lambda} \;\; \text{and} \;\; j = 1, \dots m. \tag{18}$$

Now we show that when $\text{rank}(\mathbf{X}) = p$, we can transform the lasso problem (1) into a problem of the form (17), and apply this lemma to get the desired result. First, we let $\boldsymbol{\alpha} = \mathbf{X}\boldsymbol{\beta}$ and $\mathbf{D} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$. Then the lasso problem (1) can be solved by instead solving

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\text{argmin}} \; \frac{1}{2}\|\mathbf{y} - \boldsymbol{\alpha}\|_2^2 + \lambda\|\mathbf{D}\boldsymbol{\alpha}\|_1 \;\; \text{subject to} \;\; \boldsymbol{\alpha} \in \text{col}(\mathbf{X}), \tag{19}$$

and taking $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\hat{\boldsymbol{\alpha}}$. For the original lasso problem (1), we may assume without a loss of a generality that $\mathbf{y} \in \text{col}(\mathbf{X})$, because otherwise we can replace $\mathbf{y}$ by $\mathbf{y}'$, its projection onto $\text{col}(\mathbf{X})$, and the loss term decouples: $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 = \|\mathbf{y} - \mathbf{y}'\|_2^2 + \|\mathbf{y}' - \mathbf{X}\boldsymbol{\beta}\|_2^2$. Therefore we can drop the constraint $\boldsymbol{\alpha} \in \text{col}(\mathbf{X})$ in (19), because by writing $\boldsymbol{\alpha} = \boldsymbol{\alpha}' + \boldsymbol{\alpha}''$ for $\boldsymbol{\alpha}' \in \text{col}(\mathbf{X})$ and $\boldsymbol{\alpha}'' \perp \text{col}(\mathbf{X})$, we see that the loss term is minimized when $\boldsymbol{\alpha}'' = 0$ and the penalty term is unaffected by $\boldsymbol{\alpha}''$, as $\mathbf{D}\boldsymbol{\alpha}'' = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\alpha}'' = 0$. Hence we have shown that the lasso problem (1) can be solved by solving (17) with $\mathbf{D} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ (and taking $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\hat{\boldsymbol{\alpha}}$).

Now, the solution $\hat{\mathbf{u}}(\lambda)$ of the dual problem corresponding to (17) satisfies

$$\hat{\boldsymbol{\alpha}}(\lambda) = \mathbf{y} - \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\hat{\mathbf{u}}(\lambda),$$

and so

$$\hat{\mathbf{u}}(\lambda) = \mathbf{X}^T(\mathbf{y} - \hat{\boldsymbol{\alpha}}) = \mathbf{X}^T\big(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}(\lambda)\big).$$

Thus we have exactly $\hat{u}_j(\lambda) = c_j(\lambda)$ for $j = 1, \dots p$, and applying the boundary lemma (18) completes the proof.

We note a similarity between condition (16) and the positive cone condition used in Efron et al. (2004). It is not difficult to see that the positive cone condition implies (16), and actually (16) is easier to verify because it doesn't require looking at every possible subset of columns.

A simple model in which diagonal dominance holds is when the columns of $\mathbf{X}$ are orthonormal, because then $\mathbf{X}^T\mathbf{X} = \mathbf{I}$. But the diagonal dominance condition (16) certainly holds outside of the orthonormal design case. We finish this section by giving two such examples below.

- *Equi-correlation model.* Suppose that $\|\mathbf{x}_j\|_2 = 1$ for all $j = 1, \dots p$, and $\mathbf{x}_j^T\mathbf{x}_k = \tau$ for all $j \neq k$. Then the inverse of $\mathbf{X}^T\mathbf{X}$ is

$$(\mathbf{X}^T\mathbf{X})^{-1} = \frac{1}{1-\tau}\left(\mathbf{I} \; - \; \frac{\tau}{1 + \tau(p-1)}\mathbf{1}\mathbf{1}^T\right)$$

where $\mathbf{1} \in \mathbb{R}^p$ is the vector of all ones. This is diagonally dominant as along as $\tau \geq 0$.

- *Haar basis model.* Suppose that

$$\mathbf{X} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & & & \\ 1 & 1 & \dots & 1 \end{pmatrix}, \tag{20}$$

  the lower triangular matrix of ones. Then $(\mathbf{X}^T\mathbf{X})^{-1}$ is diagonally dominant. This arises, for example, in the one-dimensional fused lasso where we solve

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^n}{\operatorname{argmin}} \ \frac{1}{2}\sum_{i=1}^{N}(y_i-\beta_i)^2+\lambda\sum_{i=2}^{N}|\beta_i-\beta_{i-1}|.$$

  If we transform this problem to the parameters $\alpha_1 = 1$, $\alpha_i = \beta_i - \beta_{i-1}$ for $i = 2, \dots N$, then we get a lasso with design $\mathbf{X}$ as in (20).

4. Strong rules for the elastic net

In the elastic net (Zou & Hastie (2005)) we solve the problem ¶

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\lambda_1\||\boldsymbol{\beta}\|_1+\frac{1}{2}\lambda_2\|\boldsymbol{\beta}\|_2^2. \tag{21}$$

Letting

$$\tilde{\mathbf{X}} = \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda_2}\cdot\mathbf{I} \end{pmatrix}, \ \ \tilde{\mathbf{y}} = \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix},$$

we can rewrite (21) as

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \ \frac{1}{2}\|\tilde{\mathbf{y}}-\tilde{\mathbf{X}}\boldsymbol{\beta}\|_2^2+\lambda_1\|\boldsymbol{\beta}\|_1. \tag{22}$$

In this (standard lasso) form we can apply SAFE and strong rules to discard predictors. Notice $|\tilde{\mathbf{x}}_j^T\tilde{\mathbf{y}}| = |\mathbf{x}_j^T\mathbf{y}|$, $\|\tilde{\mathbf{x}}_j\|_2 = \sqrt{\|\mathbf{x}_j\|_2^2+\lambda_2}$, $\|\tilde{\mathbf{y}}\|_2 = \|\mathbf{y}\|_2$. Hence the basic SAFE rule for discarding predictor j is

$$|\mathbf{x}_j^T\mathbf{y}| < \lambda_1 - \|\mathbf{y}\|_2\cdot\sqrt{\|\mathbf{x}_j\|_2^2+\lambda_2}\cdot\frac{\lambda_{1,\max}-\lambda_1}{\lambda_{1,\max}}.$$

The `glmnet` package uses the parametrization $(\alpha\lambda, (1-\alpha)\lambda)$ instead of (λ_1, λ_2). With this parametrization the basic SAFE rule has the form

$$|\mathbf{x}_j^T\mathbf{y}| < \alpha\lambda - \|\mathbf{y}\|_2\cdot\sqrt{\|\mathbf{x}_j\|^2+(1-\alpha)\lambda}\cdot\frac{\lambda_{\max}-\lambda}{\lambda_{\max}}. \tag{23}$$

The strong screening rules have a simple form under the `glmnet` parametrization for the elastic net. The basic strong rule for discarding predictor j is

$$|\mathbf{x}_j^T\mathbf{y}| < \alpha(2\lambda-\lambda_{\max}), \tag{24}$$

¶This is the original form of the "naive" elastic net proposed in Zou & Hastie (2005), with additional the factors of 1/2, just for notational convenience.

while the sequential strong rule is

$$|\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}(\lambda_{k-1}))| < \alpha(2\lambda_k - \lambda_{k-1}). \tag{25}$$

Figure 7 shows results for the elastic net with standard independent Gaussian data with $N = 100$, $p = 1000$, for three values of α. There were no violations in any of these figures, that is, no predictor was discarded that had a nonzero coefficient at the actual solution. Again we see that the strong sequential rule performs extremely well, leaving only a small number of excess predictors at each stage.

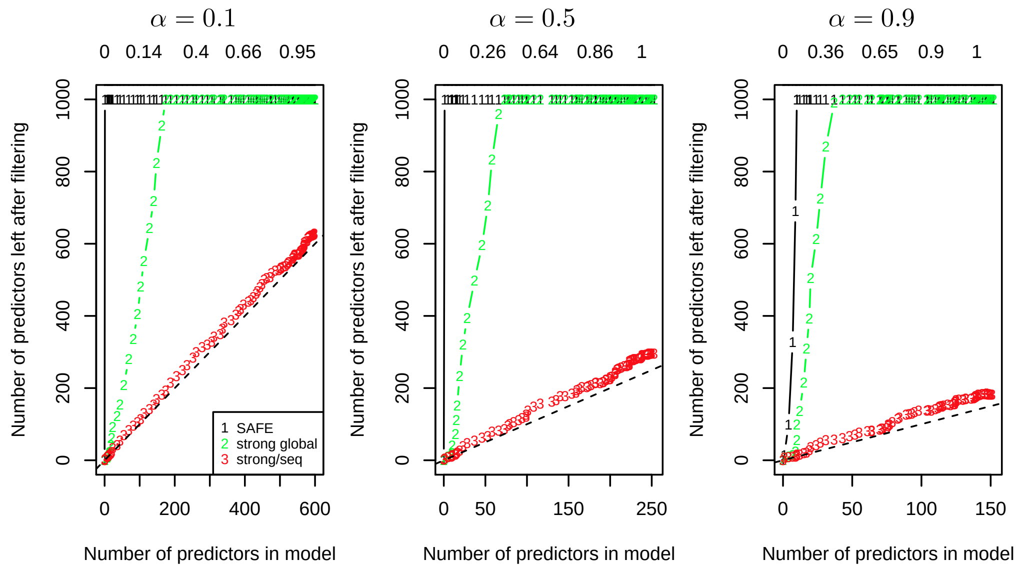

Fig. 7. *Elastic net: results for the different screening rules* (23), (24), (25) *for three different values of the mixing parameter* α. *In the plots, we are fitting along a path of decreasing* λ *values and the plots show the number of predictors left after screening at each stage. The proportion of variance explained by the model is shown along the top of the plot. There were no violations of any of the rules in the 3 scenarios.*

5. Strong rules for logistic regression

In this setting, we have a binary response $y_i \in \{0, 1\}$ and we assume the logistic model

$$\Pr(Y = 1|x) = \mathbf{p}(\beta_0, \boldsymbol{\beta}) = 1/(1 + \exp(-\beta_0 - x^T\boldsymbol{\beta})).$$

Letting $p_i = \Pr(Y = 1|x_i)$, we seek the coefficient vector $\hat{\boldsymbol{\beta}}$ that minimizes the penalized (negative) log-likelihood,

$$\hat{\beta}_0, \hat{\boldsymbol{\beta}} = \underset{\beta_0 \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \; -\sum_{i=1^n} \big(y_i \log p_i + (1 - y_i)\log(1 - p_i)\big) + \lambda \|\boldsymbol{\beta}\|_1. \tag{26}$$

(We typically do not penalize the intercept $\hat{\beta}_0$.) El Ghaoui et al. (2010) derive a SAFE rule for discarding predictors in this problem, based on the inner products between $\mathbf{y}$ and each predictor, and derived using similar arguments to those given in the Gaussian case.

Here we investigate the analogue of the strong rules (5) and (6). The KKT conditions for problem (26) are

$$\mathbf{x}_j^T\big(\mathbf{y} - \mathbf{p}(\hat{\beta}_0, \hat{\boldsymbol{\beta}})\big) = \lambda\gamma_j \quad \text{for } j = 1, \ldots p, \tag{27}$$

where γ_j is the jth component of the subgradient of $\|\hat{\boldsymbol{\beta}}\|_1$, the same as in (9). Immediately we can see the similarity between (8) and (9). Now we define $c_j(\lambda) = \mathbf{x}_j^T\big(\mathbf{y} - \mathbf{p}(\hat{\boldsymbol{\beta}}(\lambda))\big)$, and again we assume (10). This leads to the basic strong rule, which discards predictor j if

$$|\mathbf{x}_j^T(\mathbf{y} - \bar{\mathbf{p}})| < 2\lambda - \lambda_{\max}, \tag{28}$$

where $\bar{\mathbf{p}} = \mathbf{1}\bar{y}$ and $\lambda_{\max} = \max_i |\mathbf{x}_i^T(\mathbf{y} - \bar{\mathbf{p}})|$. It also leads to the sequential strong rule, which starts with the fit $\mathbf{p}(\hat{\beta}_0(\lambda_{k-1}), \hat{\boldsymbol{\beta}}(\lambda_{k-1}))$ at λ_{k-1}, and discards predictor j if

$$\left|\mathbf{x}_j^T\Big(\mathbf{y} - \mathbf{p}(\hat{\beta}_0(\lambda_{k-1}), \hat{\boldsymbol{\beta}}(\lambda_{k-1}))\Big)\right| < 2\lambda - \lambda_0. \tag{29}$$

Figure 8 shows the result of applying these rules to the newsgroup document classification problem (Lang 1995). We used the training set cultured from these data by Koh et al. (2007). The response is binary, and indicates a subclass of topics; the predictors are binary, and indicate the presence of particular tri-gram sequences. The predictor matrix has 0.05% nonzero values. Results are shown for the basic strong rule (28) and the sequential strong rule (29). We were unable to compute the basic SAFE rule for penalized logistic regression for this example, as this had a very long computation time, using our R language implementation. But in smaller examples it performed much like the basic SAFE rule in the Gaussian case. Again we see that the sequential strong rule (29), after computing the inner product of the residuals with all predictors at each stage, allows us to discard the vast majority of the predictors before fitting. There were no violations of either rule in this example.

Some approaches to penalized logistic regression such as the `glmnet` package use a weighted least squares iteration within a Newton step. For these algorithms, an alternative approach to discarding predictors would be to apply one of the Gaussian rules within the weighted least squares iteration. However we have found rule (29) to be more effective for `glmnet`.

Finally, it is interesting to note a connection to the work of Wu et al. (2009). These authors used $|\mathbf{x}_j^T(\mathbf{y} - \bar{\mathbf{p}})|$ to screen predictors (SNPs) in genome-wide association studies, where the number of variables can exceed a million. Since they only anticipated models with say $k \leq 15$ terms, they selected a small multiple, say $10k$, of SNPs and computed the lasso solution path to k terms. All the screened SNPs could then be checked for violations to verify that the solution found was global.

6. Strong rules for general problems

Suppose that we are interested in a convex problem of the form

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\ f(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{r} c_j \|\boldsymbol{\beta}_j\|_{p_j}. \tag{30}$$

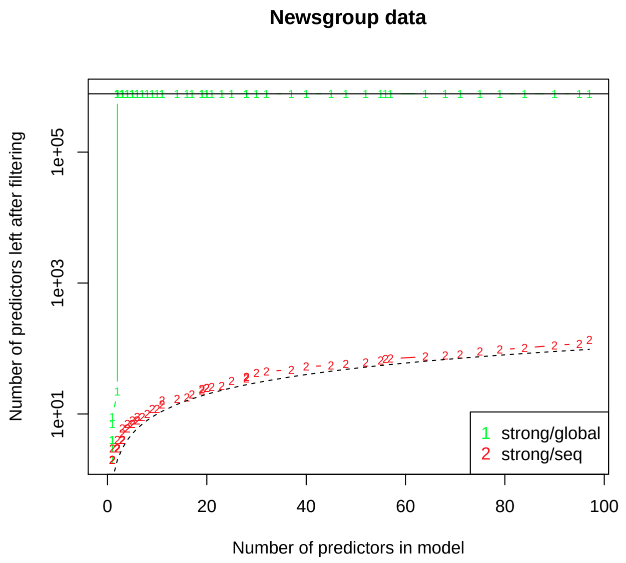

Fig. 8. *Penalized logistic regression: results for newsgroup example, using the basic strong rule* (28) *the strong sequential strong rule* (29)*. The broken curve is the 1-1 line, drawn on the log scale.*

Here f is a convex and differentiable function, and $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots \boldsymbol{\beta}_r)$ with each $\boldsymbol{\beta}_j$ being a scalar or a vector. Also $\lambda \geq 0$, and $c_j \geq 0$, $p_j \geq 1$ for each $j = 1, \ldots r$. The KKT conditions for problem (30) are

$$-\nabla_j f(\hat{\boldsymbol{\beta}}) = \lambda c_j \boldsymbol{\theta}_j \quad \text{for} \;\; j = 1, \ldots r, \tag{31}$$

where $\nabla_j f(\hat{\boldsymbol{\beta}}) = (\partial f(\hat{\boldsymbol{\beta}})/\partial \boldsymbol{\beta}_{j_1}, \ldots \partial f(\hat{\boldsymbol{\beta}})/\partial \boldsymbol{\beta}_{j_m})$ if $\boldsymbol{\beta}_j = (\beta_{j_1}, \ldots \beta_{j_m})$ (and is simply the jth partial derivative if $\boldsymbol{\beta}_j$ is a scalar). Above, $\boldsymbol{\theta}_j$ is a subgradient of $\|\hat{\boldsymbol{\beta}}_j\|_{p_j}$, and satisfies $\|\boldsymbol{\theta}_j\|_{q_j} \leq 1$, where $1/p_j + 1/q_j = 1$. In other words, $\|\cdot\|_{p_j}$ and $\|\cdot\|_{q_j}$ are dual norms. Furthermore, $\|\boldsymbol{\theta}_j\|_{q_j} < 1$ implies that $\hat{\boldsymbol{\beta}}_j = 0$.

The strong rules can be derived by starting with the assumption that each $\nabla_j f(\hat{\boldsymbol{\beta}}(\lambda))$ is a Lipschitz function of λ with respect to the ℓ_{q_j} norm, that is,

$$\big\|\nabla_j f\big(\hat{\boldsymbol{\beta}}(\lambda)\big) - \nabla_j f\big(\hat{\boldsymbol{\beta}}(\tilde{\lambda})\big)\big\|_{q_j} \leq c_j|\lambda - \tilde{\lambda}| \quad \text{for any} \;\; \lambda, \tilde{\lambda} \;\; \text{and} \;\; j = 1, \ldots r. \tag{32}$$

Now the sequential strong rule can be derived just as before: suppose that we know the solution $\hat{\boldsymbol{\beta}}(\lambda_{k-1})$ at λ_{k-1}, and are interested in discarding predictors for the optimization problem (30) at $\lambda_k < \lambda_{k-1}$. Observe that for each j, by the triangle inequality,

$$\begin{aligned}\big\|\nabla_j f\big(\hat{\boldsymbol{\beta}}(\lambda_k)\big)\big\|_{q_j} &\leq \big\|\nabla_j f\big(\hat{\boldsymbol{\beta}}(\lambda_{k-1})\big)\big\|_{q_j} + \big\|\nabla_j f\big(\hat{\boldsymbol{\beta}}(\lambda_k)\big) - \nabla_j f\big(\hat{\boldsymbol{\beta}}(\lambda_{k-1})\big)\big\|_{q_j} \\ &< \big\|\nabla_j f\big(\hat{\boldsymbol{\beta}}(\lambda_{k-1})\big)\big\|_{q_j} + c_j(\lambda_{k-1} - \lambda_k),\end{aligned} \tag{33}$$

the second line following from the assumption (32). The sequential strong rule for discarding predictor j is therefore

$$\big\|\nabla_j f\big(\hat{\boldsymbol{\beta}}(\lambda_{k-1})\big)\big\|_{q_j} < c_j(2\lambda_k - \lambda_{k-1}). \tag{34}$$

Why? Using (33), the above inequality implies that

$$\left\|\nabla_j f\big(\hat{\boldsymbol{\beta}}(\lambda_k)\big)\right\|_{q_j} < c_j(2\lambda_k - \lambda_{k-1}) + c_j(\lambda_{k-1} - \lambda_k) = c_j\lambda_k,$$

hence $\|\boldsymbol{\theta}_j\|_{q_j} < 1$, and $\hat{\boldsymbol{\beta}}_j = 0$. The basic strong rule follows from (34) by taking $\lambda_{k-1} = \lambda_{\max} = \max_i\{\|\nabla_i f(0)\|_{q_i}/c_i\}$, the smallest value of the tuning parameter for which the solution is exactly zero.

The rule (34) has many potential applications. For example, in the graphical lasso for sparse inverse covariance estimation (Friedman et al. 2007), we observe N multivariate normal observations of dimension p, with mean 0 and covariance $\mathbf{\Sigma}$. Let $\mathbf{S}$ be the observed empirical covariance matrix, and $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$. The problem is to minimize the penalized (negative) log-likelihood over nonnegative definite matrices $\mathbf{\Theta}$,

$$\hat{\mathbf{\Theta}} = \underset{\mathbf{\Theta}\succeq 0}{\operatorname{argmin}} \; -\log\det\mathbf{\Theta} + \operatorname{tr}(\mathbf{S}\mathbf{\Theta}) + \lambda\|\mathbf{\Theta}\|_1. \tag{35}$$

The penalty $\|\mathbf{\Theta}\|_1$ sums the absolute values of the entries of $\mathbf{\Theta}$; we assume that the diagonal is not penalized. The KKT conditions for (35) can be written in matrix form as

$$\hat{\mathbf{\Sigma}} - \mathbf{S} = \lambda\mathbf{\Gamma}, \tag{36}$$

where Γ_{ij} is the (i,j)th component of the subgradient of $\|\hat{\mathbf{\Theta}}\|_1$. Depending on how we choose to make (36) fit into the general KKT conditions framework (31), we can obtain different sequential strong rules from (34). For example, by treating everything elementwise we obtain the rule: $|S_{ij} - \hat{\Sigma}_{ij}(\lambda_{k-1})| < 2\lambda_k - \lambda_{k-1}$, and this would be useful for an optimization method that operates elementwise. However, the graphical lasso algorithm proceeds in a blockwise fashion, optimizing over one whole row and column at a time. In this case, it is more effective to discard entire rows and columns at once. For a row i, let $\mathbf{s}_{12}$, $\boldsymbol{\sigma}_{12}$, and $\mathbf{\Gamma}_{12}$ denote $\mathbf{S}_{i,-i}$, $\mathbf{\Sigma}_{i,-i}$, and $\mathbf{\Gamma}_{i,-i}$, respectively. Then the KKT conditions for one row can be written as

$$\boldsymbol{\sigma}_{12} - \mathbf{s}_{12} = \lambda\mathbf{\Gamma}_{12}. \tag{37}$$

Now given two values $\lambda_k < \lambda_{k-1}$, and the solution $\hat{\mathbf{\Sigma}}(\lambda_{k-1})$ at λ_{k-1}, we have the sequential strong rule

$$\|\hat{\boldsymbol{\sigma}}_{12}(\lambda_{k-1}) - \mathbf{s}_{12}\|_\infty < 2\lambda_k - \lambda_{k-1}. \tag{38}$$

If this rule is satisfied, then we discard the entire ith row and column of $\mathbf{\Theta}$, and hence set them to zero (but retain the ith diagonal element). Figure 9 shows an example with $N = 100, p = 300$, and standard independent Gaussian variates. No violations of the rule occurred.

A better screening rule for the graphical lasso was found by Witten & Friedman (2011), after this article was completed. It has the simple form

$$||s_{12}||_\infty < \lambda. \tag{39}$$

In other words, we discard a row and column if all elements in that row and column are less than λ. This simple rule is safe: it never discards predictors erroneously.

As another example, the group lasso (Yuan & Lin 2007) solves the optimization problem

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \; \frac{1}{2}\left\|\mathbf{y} - \sum_{g=1}^{G}\mathbf{X}_g\boldsymbol{\beta}_g\right\|_2^2 + \lambda\sum_{g=1}^{G}\sqrt{n_g}\|\boldsymbol{\beta}_g\|_2, \tag{40}$$

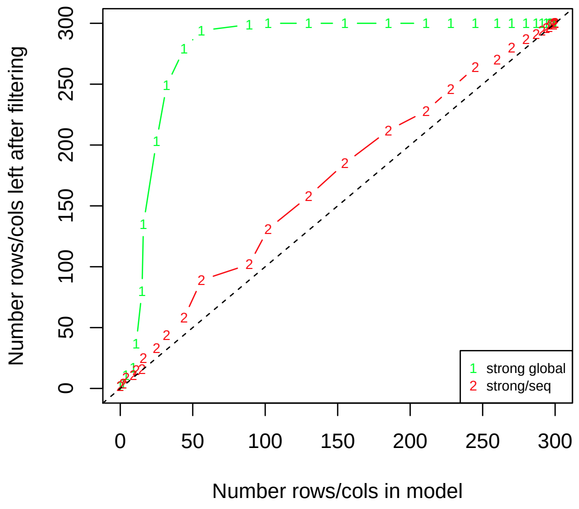

Fig. 9. *Graphical lasso: results for applying the basic and sequential strong rules* (38). *A broken line with unit slope is added for reference.*

where $\mathbf{X}_g$ is the $N \times n_g$ data matrix for the gth group. The KKT conditions for (40) are

$$\mathbf{X}_g^T\Big(\mathbf{y} - \sum_{\ell=1}^{G} \mathbf{X}_\ell \hat{\boldsymbol{\beta}}_\ell\Big) = \lambda\sqrt{n_g}\boldsymbol{\theta}_g \quad \text{for} \quad g = 1, 2, \ldots G,$$

where $\boldsymbol{\theta}_g$ is a subgradient of $\|\hat{\boldsymbol{\beta}}_g\|_2$. Hence, given the solution $\hat{\boldsymbol{\beta}}(\lambda_{k-1})$ at λ_{k-1}, and considering a tuning parameter value $\lambda_k < \lambda_{k-1}$, the sequential strong rule discards the gth group of coefficients entirely (that is, it sets $\hat{\boldsymbol{\beta}}_g(\lambda_k) = 0$) if

$$\Big\|\mathbf{X}_g^T\Big(\mathbf{y} - \sum_{\ell=1}^{G} \mathbf{X}_\ell \hat{\boldsymbol{\beta}}_\ell(\lambda_{k-1})\Big)\Big\|_2 < \sqrt{n_g}(2\lambda_k - \lambda_{k-1}).$$

7. Implementation and numerical studies

The strong sequential rule (34) can be used to provide potential speed improvements in convex optimization problems. Generically, given a solution $\hat{\boldsymbol{\beta}}(\lambda_0)$ and considering a new value $\lambda < \lambda_0$, let $\mathcal{S}(\lambda)$ be the indices of the predictors that survive the screening rule (34): we call this the *strong set*. Denote by $\mathcal{E}$ the eligible set of predictors. Then a useful strategy would be

(a) Set $\mathcal{E} = \mathcal{S}(\lambda)$.
(b) Solve the problem at value λ using only the predictors in $\mathcal{E}$.

(c) Check the KKT conditions at this solution for all predictors. If there are no violations, we are done. Otherwise add the predictors that violate the KKT conditions to the set $\mathcal{E}$, and repeat steps (b) and (c).

Depending on how the optimization is done in step (b), this could be quite effective.

First we consider a generalized gradient procedure for fitting the lasso. The basic iteration is

$$\hat{\boldsymbol{\beta}} \leftarrow S_{t\lambda}\Big(\hat{\boldsymbol{\beta}} + t \cdot \mathbf{X}^T(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})\Big)$$

where $S_{t\lambda}(x) = \mathrm{sgn(x)}(|x| - t\lambda)_+$ is the soft-threshold operator, and t is a stepsize. When $p > N$, the strong rule reduces the Np operations per iteration to $\approx N^2$. As an example, we applied the generalized gradient algorithm with approximate backtracking to the lasso with $N = 100$, over a path of 100 values of λ spanning the entire relevant range. The results in Table 3 show the potential for a significant speedup.

Next we consider the `glmnet` procedure, in which coordinate descent is used, with warm starts over a grid of decreasing values of λ. In addition, an "ever-active" set of predictors $\mathcal{A}(\lambda)$ is maintained, consisting of the indices of all predictors that have had a nonzero coefficient for some λ' greater than the current value λ under consideration. The solution is first found for this set, then the KKT conditions are checked for all predictors. If there are no violations, then we have the solution at λ; otherwise we add the violators into the active set and repeat.

The existing `glmnet` strategy and the strategy outlined above are very similar, with one using the ever-active set $\mathcal{A}(\lambda)$ and the other using the strong set $\mathcal{S}(\lambda)$. Figure 10 shows the active and strong sets for an example. Although the strong rule greatly reduces the total number of predictors, it contains more predictors than the ever-active set; accordingly, the ever-active set incorrectly excludes predictors more often than the strong set. This effect is due to the high correlation between features and the fact that the signal variables have coefficients of the same sign. It also occurs with logistic regression with lower correlations, say 0.2.

In light of this, we find that using both $\mathcal{A}(\lambda)$ and $\mathcal{S}(\lambda)$ can be advantageous. For `glmnet` we adopt the following combined strategy:

(a) Set $\mathcal{E} = \mathcal{A}(\lambda)$.
(b) Solve the problem at value λ using only the predictors in $\mathcal{E}$.
(c) Check the KKT conditions at this solution for all predictors in $\mathcal{S}(\lambda)$. If there are violations, add these predictors into $\mathcal{E}$, and go back to step (a) using the current solution as a warm start.
(d) Check the KKT conditions for all predictors. If there are no violations, we are done. Otherwise add these violators into $\mathcal{A}(\lambda)$, recompute $\mathcal{S}(\lambda)$ and go back to step (a) using the current solution as a warm start.

Note that violations in step (c) are fairly common, while those in step (d) are rare. Hence the fact that the size of $\mathcal{S}(\lambda)$ is $\ll p$ makes this an effective strategy.

We implemented this strategy and compare it to the standard `glmnet` algorithm in a variety of problems, shown in Tables 2 and 4. We see that the new strategy offers a speedup factor of 20 or more in some cases, and never seems to slow the computations substantially.

The strong sequential rules also have the potential for space savings. With a large dataset, one could compute the inner products with the residual offline to determine the

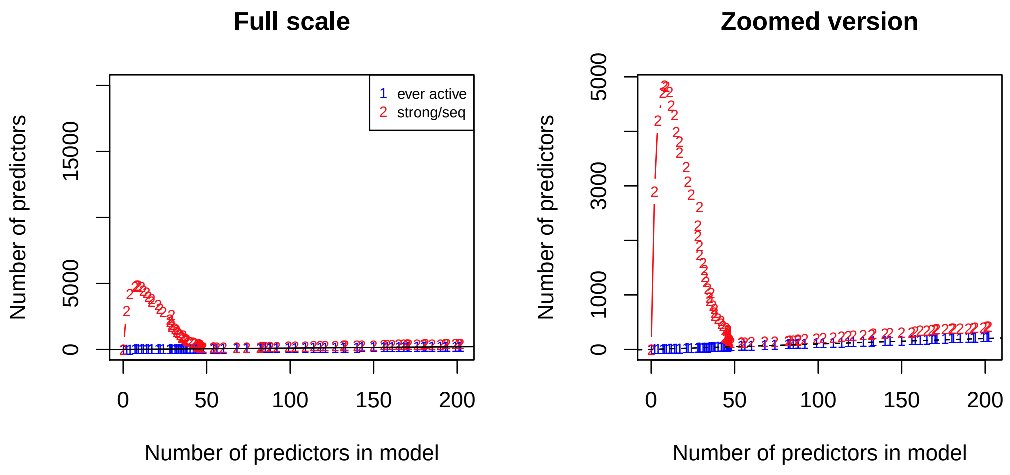

Fig. 10. *Gaussian lasso setting,* $N = 200, p = 20,000$*, pairwise correlation between features of* 0.7*. The first 50 predictors have positive, decreasing coefficients. Shown are the number of predictors left after applying the strong sequential rule* (6) *and the number that have ever been active (that is, had a nonzero coefficient in the solution) for values of* λ *larger than the current value. A broken line with unit slope is added for reference. The right-hand plot is a zoomed version of the left plot.*

strong set of predictors, and then carry out the intensive optimization steps in memory using just this subset of the predictors.

The newest versions of the `glmnet` package, available on the CRAN archive, incorporate the strong rules discussed in this paper. In addition, R language scripts for the examples in this paper will be made freely available at the url `http://www-stat.stanford.edu/~tibs/strong`.

8. Discussion

The global strong rule (3) and especially the sequential strong rule (4) are extremely useful heuristics for discarding predictors in lasso-type problems. In this paper we have shown how to combine these rules with simple checks of the Karush-Kuhn-Tucker (KKT) conditions to ensure that the exact solution to the convex problem is delivered, while providing a substantial reduction in computation time. We have also derived more general forms of these rules for logistic regression, the elastic net, group lasso, graphical lasso, and general p-norm regularization. In future work it would be important to understand why these rules work so well (rarely make errors) when $p \gg N$.

Acknowledgements

We thank Stephen Boyd for his comments, and Laurent El Ghaoui and his coauthors for sharing their paper with us before publication and for their helpful feedback on their work. We also thank the referees and editors for constructive suggestions that substantially improved this work. Robert Tibshirani was supported by National Science Foundation Grant

DMS-9971405 and National Institutes of Health Contract N01-HV-28183. Jonathan Taylor and Ryan Tibshirani were supported by National Science Foundation Grant DMS-0906801.

References

Candes, E. J. & Plan, Y. (2009), 'Near-ideal model selection by ℓ_1 minimization', *Annals of Statistics* **37**(5), 2145–2177.

Chen, S., Donoho, D. & Saunders, M. (1998), 'Atomic decomposition for basis pursuit', *SIAM Journal on Scientific Computing* **20**(1), 33–61.

Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2004), 'Least angle regression', *Annals of Statistics* **32**(2), 407–499.

El Ghaoui, L., Viallon, V. & Rabbani, T. (2010), Safe feature elimination in sparse supervised learning, Technical Report UC/EECS-2010-126, EECS Dept., University of California at Berkeley.

El Ghaoui, L., Viallon, V. & Rabbani, T. (2011), Safe feature elimination for the lasso and sparse supervised learning. Submitted.

Fan, J. & Lv, J. (2008), 'Sure independence screening for ultrahigh dimensional feature space', *Journal Of The Royal Statistical Society Series B* **70**(5), 849–911.

Friedman, J., Hastie, T., Hoefling, H. & Tibshirani, R. (2007), 'Pathwise coordinate optimization', *Annals of Applied Statistics* **2**(1), 302–332.

Fuchs, J. (2005), 'Recovery of exact sparse representations in the presense of noise', *IEEE Transactions on Information Theory* **51**(10), 3601–3608.

Koh, K., Kim, S.-J. & Boyd, S. (2007), 'An interior-point method for large-scale l1-regularized logistic regression', *Journal of Machine Learning Research* **8**, 1519–1555.

Lang, K. (1995), Newsweeder: Learning to filter netnews., *in* 'Proceedings of the Twenty-First International Conference on Machine Learning (ICML)', pp. 331–339. Available as a saved R data object at `http://www-stat.stanford.edu/ hastie/glmnet`.

Meinhausen, N. & Buhlmann, P. (2006), 'High-dimensional graphs and variable selection with the lasso', *Annals of Statistics* **34**, 1436–1462.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society Series B* **58**(1), 267–288.

Tibshirani, R. & Taylor, J. (2011), 'The solution path of the generalized lasso', *Annals of Statistics* **39**(3), 1335–1371.

Tropp, J. (2006), 'Just relax: Convex programming methods for identifying sparse signals in noise', *IEEE Transactions on Information Theory* **3**(52), 1030–1051.

Wainwright, M. J. (2009), 'Sharp thresholds for high-dimensional and noisy sparsity recovery using l1-constrained quadratic programming (lasso)', *IEEE Trans. Inf. Theor.* **55**, 2183–2202.

Witten, D. & Friedman, J. (2011), 'A fast screening rule for the graphical lasso', *Journal of Computational and Graphical Statistics, to appear* .

Wu, T. T., Chen, Y. F., Hastie, T., Sobel, E. & Lange, K. (2009), 'Genomewide association analysis by lasso penalized logistic regression', *Bioinformatics* **25**(6), 714–721.

Yuan, M. & Lin, Y. (2007), 'Model selection and estimation in regression with grouped variables', *Journal of the Royal Statistical Society, Series B* **68**(1), 49–67.

Zhao, P. & Yu, B. (2006), 'On model selection consistency of the lasso', *Journal of Machine Learning Research* **7**, 2541–2563.

Zou, H. & Hastie, T. (2005), 'Regularization and variable selection via the elastic net', *Journal of the Royal Statistical Society Series B.* **67**(2), 301–320.

Appendix: Derivation of the SAFE rule

The basic SAFE rule of El Ghaoui et al. (2010) for the lasso is defined as follows: fitting at λ, we discard predictor j if

$$|\mathbf{x}_j^T\mathbf{y}| < \lambda - \|\mathbf{x}_j\|_2\|\mathbf{y}\|_2\frac{\lambda_{\max}-\lambda}{\lambda_{\max}}, \tag{41}$$

where $\lambda_{\max} = \max_j |\mathbf{x}_j^T\mathbf{y}|$ is the smallest λ for which all coefficients are zero. The authors derive this bound by looking at a dual of the lasso problem (1). This dual has the following form. Let $G(\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{y}\|_2^2 - \frac{1}{2}\|\mathbf{y}+\boldsymbol{\theta}\|_2^2$. Then the dual problem is

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ G(\boldsymbol{\theta}) \ \text{ subject to } \ |\mathbf{x}_j^T\boldsymbol{\theta}| \le \lambda \ \text{ for } j = 1, \dots p. \tag{42}$$

The relationship between the primal and dual solutions is $\hat{\boldsymbol{\theta}} = \mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{y}$, and

$$\mathbf{x}_j^T\hat{\boldsymbol{\theta}} \in \begin{cases} \{+\lambda\} & \text{if } \hat{\beta}_j > 0 \\ \{-\lambda\} & \text{if } \hat{\beta}_j < 0 \\ [-\lambda, \lambda] & \text{if } \hat{\beta}_j = 0 \end{cases} \tag{43}$$

for each $j = 1, \dots p$.

Here is the argument that leads to (41). Suppose that we have a dual feasible point $\boldsymbol{\theta}_0$: that is, $|\mathbf{x}_j^T\boldsymbol{\theta}| \le \lambda$ for $j = 1, 2, \dots p$. Below we discuss specific choices for $\boldsymbol{\theta}_0$. Let $\gamma = G(\boldsymbol{\theta}_0)$ Hence γ represents a lower bound for the value of G at the solution $\hat{\boldsymbol{\theta}}$. Therefore we can add the constraint $G(\boldsymbol{\theta}) \ge \gamma$ to the dual problem (42) and problem is changed. Then for each predictor j, we find

$$m_j = \max_{\boldsymbol{\theta}} \ |\mathbf{x}_j^T\boldsymbol{\theta}| \ \text{ subject to } G(\boldsymbol{\theta}) \ge \gamma. \tag{44}$$

If $m_j < \lambda$ (note the strict inequality), then we know that at the solution $|\mathbf{x}_j^T\hat{\boldsymbol{\theta}}| < \lambda$, which implies that $\hat{\beta}_j = 0$ by (43). In other words, if the inner product $|\mathbf{x}_j^T\boldsymbol{\theta}|$ never reaches the level λ over the set feasible set $G(\boldsymbol{\theta}) \ge \gamma$, then the coefficient $\hat{\beta}_j$ must equal zero.

Now for a given lower bound γ, the problem (44) can be solved explicitly, and this gives $m_j = |\mathbf{x}_j^T\mathbf{y}| + \sqrt{\mathbf{y}^T\mathbf{y} - 2\gamma} \cdot ||\mathbf{x}_j||_2$. Then the rule $m_j < \lambda$ is equivalent to

$$|\mathbf{x}_j^T\mathbf{y}| < \lambda - \sqrt{\mathbf{y}^T\mathbf{y} - 2\gamma} \cdot ||\mathbf{x}_j||_2 \tag{45}$$

To make this usable in practice, we need to find a dual feasible point $\boldsymbol{\theta}_0$ and substitute the resulting lower bound $\gamma = G(\boldsymbol{\theta}_0)$ into expression (45). A simple dual feasible point is $\boldsymbol{\theta}_0 = \mathbf{y} \cdot (\lambda/\lambda_{\max})$ and this yields $\gamma = (1/2)\mathbf{y}^T\mathbf{y}(1 - (1 - \lambda/\lambda_{\max})^2)$; substituting into expression (45) gives the basic SAFE rule (41).

A better feasible point $\boldsymbol{\theta}_0$ (that is, giving a higher lower bound) will yield a rule in (45) that discards more predictors. For example, the recursive SAFE rule starts with a solution $\hat{\boldsymbol{\beta}}(\lambda_0)$ for some $\lambda_0 > \lambda$ and the corresponding dual point $\boldsymbol{\theta}_0 = \mathbf{X}\hat{\boldsymbol{\beta}}(\lambda_0) - \mathbf{y}$. Then $\boldsymbol{\theta}_0$ is scaled by the factor λ/λ_0 to make it dual feasible and this leads to the recursive SAFE rule of the form

$$|\mathbf{x}_j^T\mathbf{y}| < \lambda - c \tag{46}$$

where c is a function of $\mathbf{y}, \lambda, \lambda_0$ and θ_0. Although the recursive SAFE rule has the same flavor as the sequential strong rule, it is interesting that it involves the inner products $\mathbf{x}_j^T\mathbf{y}$ rather than $\mathbf{x}_j^T\mathbf{r}$, with $\mathbf{r}$ being the residual $\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}(\lambda_0)$, Perhaps as a result, if discards far fewer predictors than the sequential strong rule.

Table 1. *Results for sequential strong rule on three large classification datasets from the UCI machine learning repository (`http://archive.ics.uci.edu/ml/`), and a standard microarray dataset.* `glmnet` *was run with the default path of 100 λ values, in both regression and classification mode. Shown are the average number of predictors left after screening by the strong rule, (averaged over the path of λ values). There were no violations of the screening rule in any of the runs.*

Dataset	Model	N	p	Average number remaining after screening	Number of violations
Arcene	Gaussian	100	$10,000$	189.8	0
	Logistic			153.4	0
Dorothea	Gaussian	800	$100,000$	292.4	0
	Logistic			162.0	0
Gisette	Gaussian	6000	5000	1987.3	0
	Logistic			622.5	0
Golub	Gaussian	38	7129	60.8	0
	Logistic			125.5	0

Table 2. *Glmnet timings (seconds) for the datasets of Table 1.*

Dataset	N	p	Model	Without strong rule	With strong rule
Arcene	100	$10,000$	Gaussian	0.32	0.25
			Binomial	0.84	0.31
Gisette	6000	5000	Gaussian	129.88	132.38
			Binomial	70.91	69.72
Dorothea	800	$100,000$	Gaussian	24.58	11.14
			Binomial	55.00	11.39
Golub	38	7129	Gaussian	0.09	0.08
			Binomial	0.23	0.35

Table 3. *Timings (seconds) for the generalized gradient procedure for solving the lasso (Gaussian case). $N = 100$ samples are generated in each case, with all entries $N(0,1)$ and no signal (regression coefficients are zero). A path of 100 λ values are used, spanning the entire operating range. Values shown are the mean and standard deviation of the mean, over 20 simulations. The times are somewhat large, because the programs were written in the R language, which is much slower than C or Fortran. However the relative timings are informative.*

p	Without strong rule	With strong rule
200	10.37 (0.38)	5.50 (0.26)
500	23.21 (0.69)	7.38 (0.28)
1000	43.34 (0.85)	8.94 (0.22)
2000	88.58 (2.73)	12.02 (0.39)

Table 4. *Glmnet timings (seconds) for fitting a lasso problem in different settings. There are $p = 20,000$ predictors, $N = 200$ observations. Values shown are mean and standard error of the mean over 20 simulations. For the Gaussian model the data were generated as standard Gaussian with pairwise correlation 0 or 0.4, and the first 20 regression coefficients equalled to $20, 19, \ldots 1$ (the rest being zero). Gaussian noise was added to the linear predictor so that the signal-to-noise ratio was about 3.0. For the logistic model, the outcome variable y was generated as above, and then transformed to $(\mathrm{sign}(y) + 1)/2$. For the survival model, the survival time was taken to be the outcome y from the Gaussian model above and all observations were considered to be uncensored.*

Setting	Correlation	Without strong rule	With strong rule
Gaussian	0	0.99 (0.02)	1.04 (0.02)
	0.4	2.87 (0.08)	1.29 (0.01)
Binomial	0	3.04 (0.11)	1.24 (0.01)
	0.4	3.25 (0.12)	1.23 (0.02)
Cox	0	178.74 (5.97)	7.90 (0.13)
	0.4	120.32 (3.61)	8.09 (0.19)
Poisson	0	142.10 (6.67)	4.19 (0.17)
	0.4	74.20 (3.10)	1.74 (0.07)